$2.00

THE JACG NEWSLETTER

THE JERSEY ATARI COMPUTER GROUP

VOLUME 8 NUMBER 10 BBS : 201-298-0161 DECEMBER 1988

FROM THE EDITOR'S DESK

A new year (administratively) for the JACG, and a chance for stabilization and growth of the membership. The club is primarily composed of 8-Bit ATARIANS, and 16-Bit ATARIANS who have moved up? from 8-Bit ATARI computers. I see clearly a pattern wherein new ATARI owners, primarily of the 16-Bit ilk...but XE Game Machine owners also, are not joining the group. I think that this is more a function of the lack of information relating to the existence and usefulness of the JACG than that of an increased level of sophistication among the ATARI computer-buying public. Although not to the point of paranoia...it is unsettling to know that one's existence is basically unknown. I don't propose a solution...I only highlight a fact.

Mea Culpa

Lat month in the short modification listing in Neil Van Dost's article...there was a line 12...it should have been line 125. Somewhere in my editing...I edited out the 5...for those of you inconvenienced by this...I apologize (would you believe that it was intentional to see how many tried to use the modification?) Naw...I didn't think so!

SEARCHING

We are still looking for a capable person to take over the mantle of EDITOR of this august publication. If anyone is interested...please contact me...please!

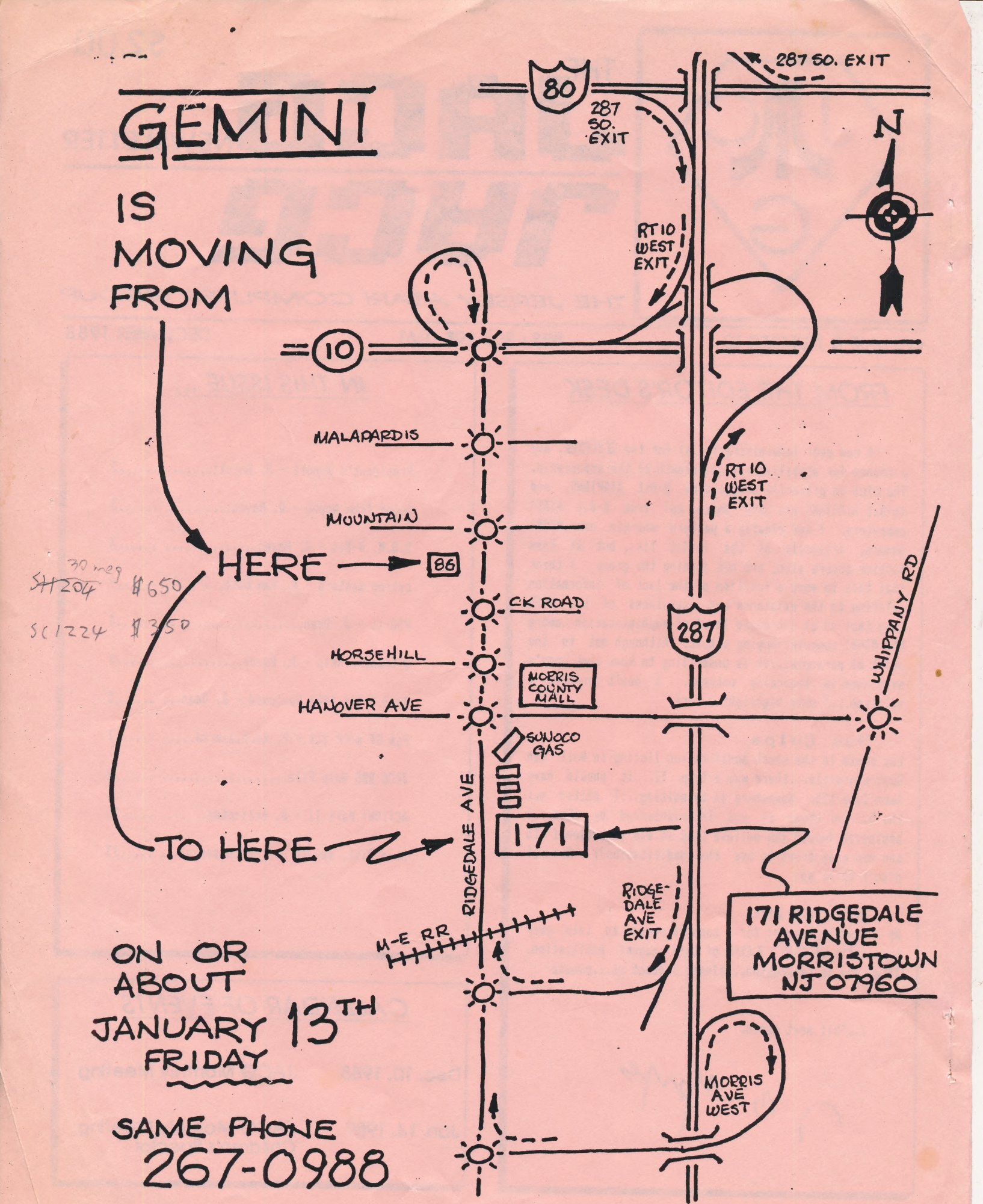
...'til next month.

D. B. Noyes

IN THIS ISSUE ...

CALENDAR OF EVENTS

Dec. 10, 1988	JACG Monthly Meeting
Jan. 14, 1989	JACG Monthly Meeting: Children's Special


GEMINI
IS
MOVING
FROM
HERE
TO HERE
ON OR
ABOUT
JANUARY 13TH
FRIDAY
SAME PHONE
267-0988
80
287 SO. EXIT
287 SO. EXIT
N
RT 10 WEST EXIT
10
MALAPARDIS
MOUNTAIN
RT 10 WEST EXIT
86
CK ROAD
287
HORSEHILL
MORRIS COUNTY MALL
WHIPPANY RD
HANOVER AVE
SUNOCO GAS
RIDGEDALE AVE
171
RIDGE-DALE AVE EXIT
M-E RR
171 RIDGEDALE AVENUE MORRISTOWN NJ 07960
MORRIS AVE WEST

PRESIDENT'S REPORT
by Gary Gorski

PRESIDENTS REPORT
Looking Ahead

Gary J. Gorski - JACG

With the holiday season just starting, I would like to take the opportunity to wish everyone a safe, healthy,enjoyable, and prosperous holiday. I will be looking forward to reading the many many reviews that will be submitted to the JACG Newsletter over the next few months. Please take the time to write your fair share of articles. You can write about that new piece of software or hardware that you received over the holidays. You can also write about an old piece, that you love or hate. Feel free to share your comments with all of us! You never know, someone else might really appreciate reading your article and getting another point of view about it, before buying it.

We will need a few children to do DEMOS at the January 14th meeting. If you would like to volunteer your child, please call me, or one of the Vice Presidents (Dave or John). As for myself, feel free to call me anytime. My number is (201) 241-4554, and I do have an answering machine on the line. Call me if you have a question, comment, complaint, or a suggestion - I WANT TO HEAR FROM YOU !!! The only way to operate this club is if we get input from the members. Speaking of members, I would like a few volunteers to set up a membership drive, as you all know, we are below 200 members for the first time. It is important, that as a team we work together to keep this club running !! PLEASE, ANY SUGGESTIONS that you might have are important, let us hear from you. I can also be reached on the JACG BBS, (201) 298-0161.

Well, looking ahead for 1989, we have January set aside as CHILDRENS month. Bring your children with you to the meeting. You just might be suprised at how much your child knows. And if your child wants to do a demo at the meeting, (8 or 16 bit), please let one of the officers know as soon as possible.

Also planned in the upcoming months will be our own Atari Safari. I will be discussing that at a later date. And hopefully by next month I will have a questionaire for EVERYONE to fill out about the JACG...

That about wraps it up for this month. Once again, to everyone, have a "HAPPY HOLIDAY", and happy computing !

NOISE FROM NOYES
by Dave Noyes

Tempus Fugit - I was looking through an old SOFTSIDE (defunct long ago) magazine - It was Volume Two, Number Twelve, September 1980. In it was an ad for an ATARI 800 computer with 16K of RAM - list price $1079.95, sale price $819.00. Additional 16K memory modules listed for $199.95 each - sale priced for $159.00 each. I don't remember ATARI's arch nemesis (COMMODORE) starting out like that. I think it was a case of market-based pricing (whatever the market would bear - in this case GREED) as opposed to cost-based pricing (price based upon the cost to maunufacture, distribute, etc.). Well, by now, it's just so much water under the bridge and over the dam.

Am getting a SUPRA 2400 baud modem for Xmas. It's about 1/4 the size of earlier 1200 baud modems - HAYES (tm) compatible too. It never fails to amaze me how more and more gets into smaller and smaller packages. Not being technically inclined - I can only gawk when people discuss IC's, VSLI, and all that super miniature, microscopic integration, and the jargon of today's leading edge electronics. Thank you...I'll remain a user and a consumer.

D.O.M.
8-Bit

Dave Noyes - JACG

This month the JACG 8-Bit Disk Library features Disk #182D. Side 1 consists of a Print Shop(tm) utility featuring the ability to delete/rename and/or copy/save icons from/to PS format/DOS 2.0 format. An icon lister/printer is also included. Transportation from/to Koala(tm)/Touch-Tablet(tm) and Micropainter(tm) is also available. Side 2 (courtesy of the Ol' Hackers) is a collection of 120 PS icons...most unique to our disk library. A bargain which should be in every 8-Bitters disk library!

Last month (and, of course, still available this month) is Disk #179D, side one of which has holiday graphics/sound from PAGE 6, the U.K. ATARI magazine. Side two contains a collection of holiday season PS icons culled from many sources...some of which lend themselves to the utilization of colored ribbons in successive passes to produce multi-colored graphics with your printer. Again, a bargain that should not be passed up.

cytron masters

neil van oost jr - JACG

CYTRON MASTERS is an SSI game that came out in 1982. Well, what am I doing reviewing it here at the end of 1988, you say? UH! Gee! Its like this, I picked up a brandy new still wrapped up in the box copy at the flea market before the meeting last month. I thought it looked familiar and when I got it home I knew it was. I had played this game quite a few years ago on a friend's APPLE. I remember it had kept me so engrossed that my beer went flat.

After I got home, the first thing I did after opening the box was to pop the disk into the drive on my 800, throw that little switch that disables my basic cartridge and fire her up. As the program was loading I gave the manual the ol' 1-3-10, you know.... I looked at page 1, glanced at page 3 and looked at the back cover. The back cover has some good info that will help you do battle. It contains a chart of how much power it will cost you to build your CYTRONS (CYbernetic elecTRONis deviceS). A mine will cost you 1 power unit and at the other end a missile will cost you 8 power units. With Bunker's, Shooter's, and Commander'S costing somewere in between the two.

The basic board setup consists of a Command Center, 4 power centers and your starting Cytron line up. Winning the game is simple. All you have to do to win is place a mine on the dot in the center of your opponent's Command Center, while at the same time keeping him from doing the same to you. This can take anywere from a half hour or so to several hours depending upon the skill of the players. If you are playing against the computer, as I do, you will eventually boost its skill when you start winning too many games. I am currently playing at the Master level and winning one out of three....well one out of four....would you believe one out of five?

The game was designed by Dan Bunten and he has truly created a battle simulation game that combines strategy thinking with arcade action. The game is fast-paced and gets your blood to perking and keeps you on the edge of your seat. Part of what makes this game exciting is the way Mr. Bunten has created a window into a different universe were you and your opponent have the ability to be in constant communication with your field armies and instruct your field commanders or even individual units on were to attack next.

Even tho this game has been out for several years it is still a winner in my book. Over the past couple of weeks it has kept me entertained for hours on end. The only other thing I can say about Cytron Masters is; "Try it, You'll like it." If you own an XL or XE a translator disk is required to make it run.

PDG-16

John H. Dean - JACG

Disk of the Month - #112

In August, 1988, Linda Peckham received a new disk named VDOS(tm), and added it to the Library as Disk #112. It was Demo'ed at the meeting by Peter Rotton, and this month I would like to lift it up as the "Disk of the Month", and tell you a little more about it.

VDOS is a Shareware Product by Marathon Computer Press, P.O. Box 68503, Virginia Beach, VA 23455. The software was developed by John B. Holder, Barbara J. Behuniak and Brian Lake. Marathon Press provides a 104 page Manual, Copyrighted, for $25.00, plus $2.00 shipping charges.

The FOREWORD of the Manual states:
"From its inception VDOS was intended to serve as an operating system automation tool. Although the ATARI ST's GEM Desktop offers many functional uses, it just falls short in some areas. That is where VDOS will aid you in the full utilization of your computer." They go on to say "Although developed with the 1040 ST, Mega ST, or Hard Disk owner in mind, those of you with 520's have not been overlooked. We could have written the floppy disk driven support out of VDOS, but that would have been very unfair to thousands of single drive ST owners."

Where does the ST's GEM Desktop fall short? The author of the Manual points out "Since GEMDOS is not a truly multitasking operating system, we must employ a technique to facilitate the automation of program maintenance, upkeep, and execution that is consistent with the existing ROM based operating environment." They did this by using the Virtual Memory (i.e. disk storage) of the ST system that is connected to VDOS to handle application storage and to house the VEXEC program spooler. This is pretty much the way that UNIX works.

This approach allows the 1040 ST owners to set up a RAM Based VDOS system, and 520 ST owners to have VDOS run on a floppy disk based system. The owners of Hard Disks may be the happiest of all since VDOS will unlock the full power of application location and execution, (no matter what the depth of directory nesting within VDOS specifications).

To quote again: "Basically, VDOS instructs the VEXEC program what application has been selected by the user, and where it is located on the system virtual memory. The activity can be likened to a foot race where there are 3 runners; VDOS is runner one, VEXEC is runner 2, and the User Chosen Application Program is runner 3. First picture yourself as the line judge pulling the trigger on the starting gun by selecting the program to run. From that point VDOS hands the baton to VEXEC. VEXEC then takes the baton for the first leg of the assigned run. Following a series of program validity checks and memory scanning,

VEXEC hands the baton over to the next runner, Your Application Program. From there the program that has been assigned and executed has total control of the MC68000 Microprocessor. When it's done and has crossed the finish line, VEXEC picks up the baton again and returns it to VDOS (since VDOS owns the processor baton). This oversimplified and humorous analogy is not meant to insult your intelligence, it is provided here for those that really don't understand how system level processes are handled. It serves the example well and was chosen over a long & dry explanation of how VDOS does its job."

ULTRASYNC HELP

Dave Davey

This file explains how to connect an Atari ST or Mega ST to to a Princeton Graphic Systems Ultrasync. This is a multisync monitor with a 12" screen, 0.28mm dot pitch, and 800*600 max resolution. Some of the features of the Ultrasync are a tilt/swivel base, adjustments for vertical position, vertical size, horizontal position, and horizontal size, an underscan/overscan capability which allows full screen display in all resolutions, and three text modes for high resolution (green, amber, cyan). I have compared this monitor side-by-side with the NEC Multisync, and find it to give sharper text and at least as brilliant color display. I highly recommend this monitor to all ST users who want to be able to display all three resolutions on one monitor. The following connections detail how to rewire the Monitor Master from Practical Solutions to a DB9 pin (female) connector for use with the Ultrasync.

Hardware Approx. Prices

PGS Ultrasync - $485
Practical Solutions MONITOR MASTER - $50
DB9 female connector
Realistic SA-10 amplifier - $30
Speaker (Radio Shack) - $10

THE CONNECTIONS

COMPUTER to ULTRASYNC
(DB9)
Pin 1 (audio) Already connected to RCA jack in Monitor Master.
Pin 2 not connected
Pin 3 not connected
Pin 4 (monochrome detect) already connected
Pin 5 not connected
Pin 6 (Green) Pin 2
Pin 7 (Red) Pin 1
Pin 8 not connected
Pin 9 (Horizontal Sync.) Pin 4
Pin 10 (Blue) Pin 3
Pin 11 (Monochrome signal) Pins 1,2,3
Pin 12 (Vertical Sync.) Pin 9
Pin 13 (Ground) Pin 6

Rewire the Monitor Master so that when the button is in the OUT position, the mono detect is grounded, and the monochrome signal is switched with the RGB signals so it will be sent over all three lines. With the button IN, the RGB signals are switched on and the monochrome signal is off, and the mono detect is not connected. Wire the output to the DB9 female connector. The Ultrasync comes with a cable which has a DB9 male connector. Also, 68 ohm 1/4 watt resistors must be inserted in lines 6,7, and 10 from the ST (the RGB analog signals) to reduce the intensity. No resistor is required for the monochrome signal. Also, make sure that the signal ground and all shields are connected together. For audio, I connected the Monitor Master with a 10 watt amplifier with a $10 speaker from Radio Shack. And that's it. Good luck!

If you have any problems just leave me a message and I'll get back to you as soon as possible.

Dave Davey 73357,645

ed. note - As with all hardware modifications, the JACG, author and editor take no responsibility for any damages, etc. arising from modifications.

CAUTION

You do this modification at your own risk, both to you and the equipment.

JACG 8-BIT 1988 REVIEWED

John H. Dean - JACG

From all sides we hear or read that "The 8-bit Ataris are dead. New software is hard to find. No one is writing programs for the Atari 400-800 line." Not so! Both Commercial and Public Domain programs are being produced, and making an impact.

Let's take a look at all the new or improved material presented to the JACG in the last twelve months, not necessarily in chronological order.

First - Dot Magic and Daisy Dot II. Without going into a history of Dot Magic and Daisy Dot, suffice it to say that we now have both in our PD library and a raft of supporting fonts. What a pleasure to see our 9 pin dot matrix printers pouring out those nearly perfect letter characters in assorted type faces - with and without serifs, and fancy fonts galore. I haven't counted, but there must be more than seventy different named fonts in the public domain.

Next - Textpro. Originally released into the public domain by MIKE COLLINS and RONNIE RICHE, this is probably the ultimate word processor for the XL/XE Atari. It accommodates almost every DOS, and is particularly good with SPARTA DOS 3.2. Mary Russomano's two part tutorial of TEXTPRO version 1.2 with Extension 2.5e, beginning in the July issue of the JACG Newsletter, is outstanding. Read it, and get our library disk #147D, complete with Docs.

MOving from the world of text and word processing and approaching that of Desk Top Publishing, 1988 saw a large increase in the number of Print Shop Icons being made available, and their conversion from and to other formats. Dave Noyes edited and compiled over a hundred of them for the Holiday season rushing upon us. Dave Dvorin's review of "The Official Print Shop Handbook" in the July Newsletter shows that this area of the 8-bit world is alive, and growing.

As for the world of Desk Top Publishing, We have even seen new programming here. This year we saw the addition of Newsroom, by Springboard, to the already available 1987 News Station and Companion by Reeve Software. And they say Atari is only for games!

During 1988, Neil Van Oost Jr. has kept us up to date in the pictorial graphic world with his articles on DIF and RLE picture formatting, and the wonderful world of fractals and other computer generated graphics. Library Disks #'s 176, 177 & 179 contain over 60 pictures, and programs to show them that Neil has downloaded from Compuserve. He notes that there are most of the 4000+ files that are on Compuserve can be assessed through either RLE of the DIF format.

In the world of telecommunications, AMODEM 7.5 was added to the list of public domain software. Some say it is even better the 850 EXPRESS. It certainly is a full featured program for the Atari 8-bit modems. Look at Library Disk #132 for this one.

Getting into the field of Utilities, RAINBOW DOS, Disk #140D, was added to our Library this year. This DOS can load character sets, Micro Paint or Koala files, and has a HELP file for you. This disk also includes a pinball game called "The Black Hole" from Pete Fazio of B.A.S.I.C..

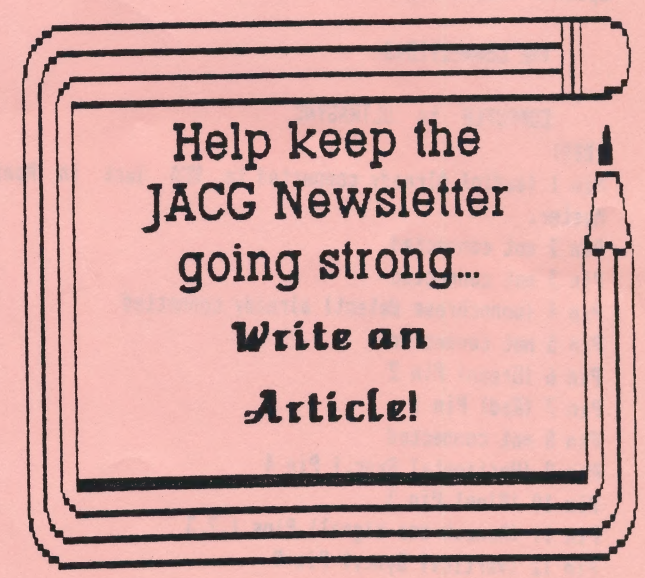
And while we are writing about DOS, we should not forget Kris Holtegaarde's lively demonstration of SPARTA DOS. Not that SPARTA DOS is new, but his demo was terrific. I had to stop off at Gemini on my way home from the meeting to buy a copy. And what is new is the fact that it is coming out on a cartridge. That's for me!

Also in the Wordprocessing world this year has been the introduction by Atari of the long awaited 80 column interface, the XEP 80. While we are still waiting for Atariwriter 80, we do have 80 columns available in TURBOWORD from Micromiser Software of Orlando, Florida. Shree Vandenberg gave us an excellent Demo of the interface and the software at our November meeting.

For the future, I am sure we can look forward to the further development of the so-called GEM Jr. for the 8-bits. "So-called" because what we have so far is disk, available at Gemini, from REEVE Software named Diamond OS. At various times we have read about Merril Ward, USA Media, and/or Alan Reeve developing a program to mimic the ST Desktop on the Atari 8-bits. This program is to be incorporated in a cartridge at some time in the future, maybe. The order form, included with the purchase of the disk, is headed "ORDER FORM - ST, Jr. DIAMOND OPERATING SYSTEM", and offers DIAMOND OS:, DIAMOND WRITE:, DIAMOND PAINT:, and OTHER: at $29.95 each, plus $1.50 for postage. A ST.,jr MOUSE is offered at $59.99. The disk, as it stands, doesn't do very much. A directory index shows, in addition to an autorun.sys of 127 sectors, a Desktop.app of 102, a Koala.drv, Mouse.drv and Touchtab.drv, each at 4 sectors, a Paint.app of 66 and a Write.app of 20 sectors. This looks promising, but that is all I know at present.

But there is more to come in 1988. I have already received an announcement from Clearstar Softechnologies at P.O. Box 140, Harrels, NC 28444, that Lightspeed C has been upgraded to version 3.0. Kitrick Sonesen asks the question in the November issue of The Access Key "Is LIGHTSPEED C worth your cash? Well, it's compatible with SpartaDOS, supports RAMDISKS of all persuasions, includes RUNTIME packages for itself, Action!, and MAC/65, and comes with a command line disk operating system of its own: Lightspeed DOS."

And lurking just around the corner is the Turbo-816 hardware, that will add 16-bit performance to your 400/800/XL/XE ala the AppleGS. This is from DataGue Software, P.O. Box 134, Ontario, Ohio, 44862.



Programming the ST with Personal Pascal – III

Paul Machiaverna - JACG

As you saw, I did not submit an article for the November issue of the JACG newsletter about Personal Pascal. The reason for this is because I wasn't happy about the way the articles were appearing on the printed page. The curly braces, { and }, did not print out on the wordprocessor used to generate most of the articles seen in this newsletter. And the listings of the Pascal source codes were not in the structured alignment that they should have been. So, I decided that it would be better to use my own Mega ST system to generate a hardcopy of the articles and send them to the editor, Dave Noyes. I am using the Timeworks Desktop Publisher ST along with a Hewlett Packard DeskJet printer to generate all my articles. As you can see, this printer yields excellent results at 300 dot per inch! Well, enough of that. Let's get to Programming the ST with Personal Pascal(PP).

This month we will be looking at the options available to the PP programmer who wishes to bypass the GEM interface and use the TOS, PC like interface. While GEM programs are nice and easy to use, programming them takes a lot of careful thought and planning. In order to use anything other than the simple Alert boxes in GEM (discussed in Part Two of this series), you must be prepared to write a considerable amount of code. If you compare the source codes of a GEM and TOS program which perform the same exact function, you will see that the GEM program is usually much longer than it's TOS counterpart. This is not to say that I think it's a waste to spend the extra time writing GEM applications if you can achieve the same end product with a TOS program. It is just that sometimes you will wish to write a program to compute a function without worrying about complicated GEM screen formatting.

First and foremost, when writing any TOS program we must tell the compiler and linker of PP that we are doing so. This is done from the PP manager screen (the one the program boots into) by selecting 'Compile for TOS' and 'Link for TOS'from the 'Options' drop down menu. A program compiled for TOS will end with TOS as it's file extender (ex. PROGRAM.TOS). When executed it will clear the GEM Desktop and show a blinking cursor in the home position of the screen while the program loads. The original 'Hello, World' program shown in part one of this series was a TOS program. Running that program will show you how the GEM Desktop is wiped clear from the screen when running a TOS program.

Compiling for TOS is what you want to choose when learning Pascal from a text book and you want to try out the source codes in them. Anytime you don't want to use GEM in your program, choose this option. All Read, Write, Readln and Writeln Pascal commands are easily used with the TOS interface. But, in using these simple Input/Output commands, you will quickly wonder how to format a screen. The Read and Write procedures alone are not easy to use when you need to access a specific location on the screen. This is where PP comes to the rescue.

Just like we used the 'GEMSUBS.PAS' file to allow access to the GEM functions, there is another file included with PP called 'AUXSUBS.PAS' which allows us to access special features such as TOS screen formatting. A sample of a very simple TOS program which includes some of the TOS procedures is shown in figure one.

Type in and save a copy of the source code shown in figure one. Then compile and run it to see what the screen format looks like. We just have used a few of the procedures available to the TOS programmer. Let's discuss them.

ClrScrn is a very simple to use subroutine which clears the screen. It should be called by any TOS program which requires a clear screen. Note that when a TOS program is run from the GEM desktop, the ST will automatically clear the screen and display a blinking cursor. However, when any program is run from a command line interpreter (such as DOS Shell by Michtron), the ST will not clear the screen unless a batch file or your program specifically requests it. This way you have more control over the program.

GotoXY is just like the Position statement in 8-bit Atari BASIC. It simply positions the cursor at the given X and Y coordinates. All Reads and Writes to the screen will be at this position. Remember that the upper left hand corner cursor position is 0, 0 in XY coordinates. In High and Medium

```
{ Simple program using some TOS procedures - Compile for TOS
Written by Paul Machiaverna JACG - November 9, 1988 }
Program Simple ;
{$I AUXSUBS.PAS} { include the TOS procedures }
Begin
  ClrScrn ; { clear the screen of the ST }
  GotoXY(10,2) ; { position the cursor at the given coordinates }
  Write('Simple formatted text on the screen') ;
  GotoXY(10,4) ;
  InverseVideo ; { make text written to screen in inverse mode }
  Write('More text in inverse video') ;
  GotoXY(20,10) ;
  NormVideo ; { make text written to screen in normal mode }
  Write('Back to normal mode text. Press a key to exit...') ;
  Repeat Until KeyPress ; { we saw this in the first part of
these articles }
  Curs_off ; { turn off blinking cursor before ending program }
End.
```

Figure One

resolution screens the lower right hand corner is 79, 24. On the Low resolution screen it is 39, 24. In PP, the X and Y coordinates must be given as a Short_Integer expression. You can give these numbers as a variable, constant, or the numbers themselves as shown above.

InverseVideo switches the background and foreground colors so that all text printed AFTER this command will be in inverse. Text on the screen will not be affected. NormVideo is the reverse of this command which uses the original settings of the background and foreground colors.

Curs_off turns off the cursor on the ST. You should always call this subroutine before ending any TOS program. Or else you take the chance of having a blinking cursor on the screen after your TOS program ends and you return to the GEM desktop. You may even call this function anytime you feel the screen would look better with the cursor turned off. The reverse of Curs_off is Curs_on and simply turns the blinking cursor on.

Some other useful TOS procedures are shown in the PP manual and make screen formatting a snap. Curs_Home will place the cursor at the upper left hand (home) position of the screen without affecting any of the text on the screen. Clr_EOS will clear text from the cursor position to the end of the screen. Curs_up, Curs_Down, Curs_Right, and Curs_Left all will move the cursor around the screen without affecting any text on the screen.

Two very useful procedures are TextColor() and TextBackground(). They allow you to set the screen colors within a TOS program. It is obvious what each affects. The number to be passed to each is a number from 0 to 15. These are the sixteen color registers available on the ST. However, remember that only 4 are available in Medium resolution and you can only choose black or white in High resolution.

There are quite a few more procedures and functions available in the AUXSUBS.PAS file. The PP manual explains them all and I suggest that you take a look at them. Experiment using them in your programs. I rather not take the time or the space here showing TOS program source codes because they are very simple to write. Anyone with a basic knowledge of Pascal should have no problem using them in their programs. I simply wanted to make them known to ST programmers who wish not to use GEM, yet are looking for easy ways to format a TOS screen.

A couple of notes about using the AUXSUBS. Some of the procedures contained in it can be used in any program(GEM, TOS, etc.). Others, such as all the screen formatting routines, can only be used in TOS or TTP programs. The screen used in TOS programs follow the VT-52 protocol. VT-52 is a type of computer terminal and is simply defined as a device for interaction between a used and a computer system. The input is via the keyboard and the output is via the screen. Therefore, it is very easy to write modem software for the ST using the TOS/VT-52 screen.

The second note about TOS programs refers to what users look for in software. If you are going to write programs for people other than yourself to use, whether it be public domain or commercial, beware than many programs are judged by the screen format. ST users favor the GEM interface. Use it if you intend to get recognition for your programming efforts easier than if you use the TOS format. ST users want easy to use programs and usually judge this ease by the use of GEM.

That's enough for TOS programming for now. Next month, ahem, next year I will return to GEM programming with a look at Dialog boxes. I will dedicate at least two articles to them because of their complexity. I wish you all a very happy and healthy holiday season with all the best for the new year. Happy Atari ST programming!

Isn't it time that you wrote a News Article?

The JACG BBS Help File

Long(er) Version,
6/30/87

This is the newly updated version of the JACG BBS Help File. Capture this file to disk, and print it out for future reference.

The structure of the JACG Bulletin Board System is subject to change without notice. In the future, look for separate Help files for each Menu on the BBS.

General Information

The JACG BBS is run on an Atari 520ST computer using BB/ST software by QMI, Inc. BB/ST is very flexible, which means that the system can be tailored to look exactly the way _we_ want it to. Therefore, if you have any suggestions for enhancements or improvements, please voice them!

Hitting your space bar or typing ctrl-S at any time causes the system to pause whatever it's printing so you can read it at your leisure. (Hitting any key will restart printing) <esc> or ctrl-C STOPS (cuts off) printing immediately and returns you to the nearest menu prompt.

Typing a question mark (?) at any menu prompt will give a list of commands for you to choose from. All commands are one letter, and you do NOT have to hit Return.

Typing a "Y" (Yell for Gary) at any menu prompt will page Gary Gorski. (Tom Shoosmith and Paul Machiaverna are Remote SysOps.) If Gary is in, he may break in to "chat" with you online. If not, the system will give you a chance to leave him a Note. (Actually, all of the SysOps read the Notes.)

Typing a "G" (Goodbye) at any menu prompt will end your session on the BBS. You will be given the opportunity to leave a Note for Gary, and then will be asked if you really want to leave. If you say yes, the BBS will hang up on you. Please use this command to leave the BBS, do NOT hang up on the board yourself!

The "&" character (ampersand) is the Fast Logout command. Typing it at any menu prompt will end your session on the BBS immediately, no questions asked. Use with care!

If you ever receive a "(more)" prompt, hit Return to print ONE more line, or SPACE BAR to print another screen. Most (more) prompts can be turned off by selecting (V)iew Options from the Main Menu, and following instructions.

If you are asked a yes/no question, hitting just Return will select the capitalized default letter. For instance, if you were asked:

Do you understand now (Y/n)?

hitting Return would select "Yes." Some other questions that the BBS asks you give you a choice of letters to type (like A/b/c). The default is, again, capitalized.

The Bulletin Menu

You arrive here first, after you log on and give the system your password. You can also get here by typing "B" from the Main Menu. Type a number to read a bulletin, or "G" to get to the Main Menu.

The Main Menu

This is the "center" of the BBS. From here you can get anywhere else on the board, and typing "G" (for Quit to Main Menu) anywhere gets you back here. (If you're at the Main Menu already, typing "Z" does a Zip Message Scan, see below.) Some of the commands available from the Main Menu are:

C - Lets you see the names of the most recent callers of the BBS.
E - Lets you send private Electronic Mail to anyone on the BBS.
N - Lets you send a Note to Gary and the other SysOps.
R - Lets you read your Email. Also lets you read the Email you've sent to others, which you may delete.
T - Shows the current time and date.
U - Prints a list of all the users of the BBS. Use this list to find the correct spelling of a person's name, so that you can send him/her Email.
V - Presents a list of statistics about yourself, and then lets you change some options, like number of lines on your screen, number of columns, whether or not you want (more) prompts, etc.

The File Area

Here's where we keep all of the goodies! Hundreds of files are waiting in the File Area for you to choose from.

To get to the File Area, type "F" from the Main Menu.

The files are grouped into several sub-areas by file type and computer type. The 8-bit areas are for Atari 400, 800, XL, and XE owners, and the 16-bit areas are for Atari ST owners. Use the "C" command to Change the sub-area you're in. The "L" command will let you List the files in the current sub-area, in either a short, concise format, or an expanded format with descriptions of each file.

Use the "D" command to download the file you want. You may refer to files by either name or number. You also have a choice of file transfer protocols: straight ASCII, Xmodem, Xmodem with CRCs, and Ymodem. To ensure error-free transmission, use Xmodem or Ymodem, whichever your terminal program supports.

You may also upload public-domain files to the BBS ("U" command). Doing so will gain you more time on the system, after the files are cleared for availability by the SysOps.

The Message Area

Electronic Mail (Email) is for private messages, but the Message Tree is for public discussions, questions, advertisements, and much more. The Message Tree is probably the most confusing part of the BBS for first-time users, also. You get to the Message Area by typing "M" from the Main Menu prompt. From here, you can read messages linearly with the (R) command, or just read the new messages (that is, the messages you haven't read already) with the (N) command. Typing "Z" from the Main Menu is the same as typing "N" from the Message Area; it's a (Z)ip quick way to read the new messages as soon as you log onto the BBS.

To actually use the tree to find or post about a topic of interest, you must type numbers. As an example, let's suppose you want to tell people about the super new word processor (Deluxe WriteItAll) that you just bought for your Atari 65XE. First, you type "M" from the Main Menu. You are brought to the Message Area, and the Top of the Tree is shown. See the list of branches, each with a number in front of it? Naturally, you want to go to the "8-Bit Branch," so type the appropriate number (3, I believe), and hit Return. (Yes, this is the one place where you DO have to use Return...)

Now you are shown the 8-Bit Branch. There is probably a list of more branches here, too. These are the various conversations about 8-bit Ataris. You're going to start another one.

Type "E". The BBS asks, "Enter a message under '8-Bit Branch'?" Answer yes, and type your message! Hit <esc> (or three RETURNs) to finish, and then type "S" to Save (or "Q" to Quit without saving) your message on the Tree.

By the way, the title of your message can be up to about 100 characters long, so sometimes your entire message can conveniently fit in the title!

The Other BBS Men

Get here by typing "O" from the Main Menu, then type "V" to view our growing list of other Bulletin Board Systems. Also check out the list of PC Pursuit-accessible BBSs; type "P".

If your favorite Bulletin Board System is not on the list, why not add it yourself? Use the "A" command, and give as much helpful information as you can. Be sure to add (PCP) to the BBS name if you know it's PC-Pursuit accessible.

The Polling Area

"V"ote in a Poll, or just "L"ook at the results. There are many public polls here on a variety of subjects. "P" from the Main Menu gets you here, and "Q", of course, gets you back.

(updated 2/3/88)

----------End of Help File----------

ACTION! FOR BASIC PROGRAMMERS

Part II: Variables

Dave Arlington - JACG

One thing I would like to say right from the start. Even if you are not interested in learning ACTION!, since we are going to learning it from a BASIC standpoint, you can learn some things about BASIC if you read this column each month. (At least I hope so!)

Onward we go. Let's start with a little race. A very simple one. I know there are all kinds of benchmark tests out there but these are just simple loop and add numbers. Get out your stopwatch.... Naw.. Let's let the computer do it for us. That's why we have one, right? Here are three sample programs. Essentially the same thing, the first is written in ATARI BASIC, the second in TURBO BASIC, and the last in ACTION! Type them in, run them, and see what you get. Remember, a jiffie is 1/60th of a second.

LISTING ONE: ATARI BASIC

```
10 POKE 20,0:POKE 19,0
20 SUM=0
30 FOR X=1 TO 250
40 SUM=SUM+X
50 NEXT X
60 PRINT "The sum is ";SUM
70 PRINT "That took ";
80 PRINT PEEK(20)+PEEK(19)*256;
90 PRINT " jiffies"
```

LISTING TWO: TURBO BASIC

```
10 Dpoke 19,0
20 Sum=0
30 FOR X=1 TO 250
40   Sum=Sum+X
50 Next X
60 Print "The sum is ";Sum
70 Print "That took ";
80 Print Peek(20)+Peek(19)*256;
90 Print " jiffies"
```

LISTING THREE: ACTION!

```
PROC main()

  CARD sum=[0], time

  BYTE ctr, t1=19, t2=20
```

```
t1=0 t2=0
FOR ctr=1 TO 250
  DO
    sum==+ctr
  OD
PrintF("The sum is %U%E", sum)
Print("That took ")
time=t2+t1$256
PrintC(time)
PrintE(" jiffies")

RETURN
```

All done? Back? OK, if your tests came out like mine, you found out that in ATARI BASIC that little program took 84 or 85 jiffies or about a second and a half. In Turbo BASIC it goes in about 39 jiffies or 2/3rds of a second. For the heck of it, compiled Turbo BASIC does it in 15 jiffies or 1/4 of a second. In ACTION! it takes 3 or 4 jiffies or faster than 1/10th of a second. That's about 21 times faster than ATARI BASIC in just this short program. In much longer programs the speed discrepancy is much greater. For a last little experiment, go back to the first listing up above in ATARI BASIC and add REMark statements with line numbers 31 through 39 and 41 through 49. Go ahead and run it, I'll wait. (and wait. Sorry, just a joke!) OK I get 137 jiffies or over 2 seconds or about 34 times slower than ACTION! on this short program. By the way adding a hundred REMarks to an ACTION! program doesn't change the speed one bit. So why is ATARI BASIC so slow and ACTION! so fast? That's what we'll discuss this month.

The first difference and the big one that showed up when we added the REMark statements to Listing One is that ATARI BASIC (and Turbo) is an interpreted, interactive language whereas ACTION! is a compiled language. What that means is this: One of BASIC's great strengths is that you can type a few lines, RUN your program, hit the break key key, maybe print out some variable values, make some changes and then try it again. You can type in commands without a line number, hit return and watch them execute. This is a great advantage sometimes when we are writing a program. But hopefully, a program will be run a lot more times than it is written and it is here we pay the price for the ease of the interactive approach.

The reason we can execute program lines directly and jump in and out of our program so quickly is that BASIC interprets every line as it comes across it, into instructions the machine can understand. In a program like we wrote above, when BASIC goes through that loop 250 times, it has to translate that line of addition into something the machine can do, 250 times. When we add the REMarks, it slows down more, because the computer doesn't remember that the first time through the loop that these were REMark statements that don't get executed. It still has to stop and figure out what they are every time through and on every REMark. So not only is BASIC an unstructured language as it is, it punishes you if you put remarks in to make your program more understandable!!

Compare this approach with how a compiled language like ACTION! handles a program. You write it in a section called the editor, go to the compiler and compile it, and then you can finally run it. Sounds like a lot of work that isn't involved in BASIC. And it is a bit more work than BASIC in the writing stage. However, when it comes time to run it, we are more than offset for the extra time we spent writing it.

What ACTION! or any other compiled language does is translate the program statements only ONCE into a machine understandable form. That is what the compiler does. Now it knows that 250 times it has to do an addition and doesn't have to stop and figure out that it is an addition statement every time. As for the REMarks, you can add a hundred of them becuase the compiler throws them away as soon as it sees them the first time.

Most compiled languages other than ACTION! are very bulky on the ATARI becuse the editor and compiler and any other subprograms you need, have to be loaded from disk. ACTION! makes writing a program almost as fast as BASIC. The editor and compiler are both in the cartridge so the speed to go from one to the other is there. From the ACTION! Monitor (where you call the compiler from), you can execute program lines and print out variable values in immediate mode just like you can in BASIC. You can insert a Break() or use the regular break key when your program is running to jump back to the monitor to do these things. If you make a typing mistake or put in something ACTION! does not understand (commonly called a syntax error) the compiler will halt, tell you what the mistake is, and when you instantly jump back into the editor, the cursor is conviently placed somewhere near the offending statement. In all, it has almost all the convenience of BASIC without the speed overhead later on.

Let's move on to variables in ACTION!. In BASIC, you have two or three basic types of variables: numeric, numeric arrays, and string variables consisting of charcaters. In ACTION! there are many types of variables and we'll be getting into some of the many types later on. For now we'll concentrate on the numeric or number holding variables.

In ACTION!, like many other structured languages, you have to declare all variables before you use them. What we mean by this, is we tell ACTION! what the name of the variable is, what type of information it will be holding, and how much memory to set aside for that variable before we can use it in our program. Many BASIC programmers groan at this percieved lack of freedom. They are used to dreaming up variables out of thin air anytime and anywhere they want to use them in a program. However, especially if you program in BASIC on an ATARI, you have been declaring variables for a long time and probably never knew it!

Think about it. Don't you have to tell BASIC the name of a string variable and how much memory to set aside for it before you use it? You just can't say A$="hello" without first DIMensioning (or declaring) A$ first. In BASIC, you can DIMension a string variable anywhere as long it is before the first time you use it. In ACTION! we don't have to worry about making sure it comes before we use the variable since we declare them first, right after the line that has the word PROC in it.

Let's look at our BASIC example in Listing one and explain another reason why BASIC is so slow. If you look at our loop, we use the variable X to count from 1 to 250. If you've ever studied anything about computers and how their memory is set up, you might remember that the computer needs only one memory location (or byte of memory) to hold any number from 0 to 255. You might think that since we never go over 250 in our loop that BASIC only uses one memory location to store X. However, since BASIC never knows when you might change that variable to something else that needs more memory locations (like a 9 digit number for instance), it has to assume the worst case.

BASIC uses not one, but 6 memory locations for every numeric variable, even if it is normally only a one byte number (0-255). (A side note here: Since BASIC always assumes the worst case, that is why ATARI BASIC is the only BASIC that makes you DIMension string variables. On the ATARI, a string variable can be as long as memory. Since there would be no room for a program if you had a string as big as memory, BASIC makes you tell it how big the string variable will be so it can figure out how much room will be left for your program. In APPLE or COMMODORE BASICs, strings can only be 255 characters maximum and so those BASICs can assume the worst if you don't dimension them, without having to worry about running out of memory for your program. OK, end of side note!) So in our example program, BASIC wastes five bytes of memory for X every time through the loop.

The value SUM only reaches 31,000, something which your computer could hold using only two memory locations (Two memory locations can hold a number as big 256 times 256 or 0-65,535). So even with a number as big as 31,000, BASIC is still wasting a four memory locations. Not only that, but when we go to add them together in line 40, we have to add two 6 byte numbers instead of adding a one and two byte number. If you stop to convert that to binary, BASIC has to add numbers holding 48 ones and zeros together every time we come to line 40. Try writing 48 ones and zeros over each other and adding them together. Now you get another idea of why BASIC is so slow, just doing simple math.

In ACTION!, things are different. If we know ctr (the ACTION! version of X in our Listing Three above) is only going to take up one byte, we tell the compiler so right at the start so it does not waste a lot of space storing it or a lot of time when it comes time to add the numbers together.

There are three types of ACTION! variables we'll learn about this month. The first one is called BYTE and it can hold (suprise!) a number from 0-255. The second type is called CARD and it can hold a number from 0-65535. It takes two memory locations. The third and last is called INT for integer and it can hold a number from -32767 to 32767. It also takes two memory locations. Maybe like I did when I first saw ACTION!, you're saying to yourself, "That seems pretty limiting. Only one type of variable can hold negative numbers and none of them can hold a real number." (A real number is a number with a decimal or fractional part.) Stick with me, sahib, and we shall see this is not so limiting as you might first think.

How do we declare a variable in ACTION!? There are three ways and they are all outlined in the sample program above. You must declare them right after the line that has the PROC on it that marks the start of the subroutine. You can have as many lines of variable declarations as you want as long as they all come before the start of the statement section. If Listing Three above we have two lines of variable declarations before we get into the statement section. You can mix them up however you like. You can declare some BYTE variables, then some CARDs, then do some more BYTEs etc.

The first way of declaring a variable is simply to tell ACTION! what type it is, (BYTE, CARD, or INT) and give it a name. You can declare more than one at a time by putting them on the same line seperated by commas.

FOR EXAMPLE:

```
BYTE ctr, rate, time, dist
```

would declare 4 BYTE variables called ctr, rate, time, and dist that could be used by your program. When you start they have no value. You must give them a value in the statement section of your program.

The second way of declaring a variable is to tell ACTION! what type (BYTE, CARD, or INT) it is and also give it an intial value by putting the initial value in brackets.

FOR EXAMPLE:

```
CARD sum=[0], hiscore=[10000], scrn
```

Continued Next Month...

Hard Disk Drive Users - BEWARE!

Paul Machiaverna - JACG

A hard disk drive is probably the most important addition that can be made to any computer system. It is many times faster than a floppy disk drive, stores incredible amounts of data, and gives instant access to software libraries without searching through floppy disk files. However, it can be dangerous to keep a lot of data in one place. Suppose the drive suddenly crashes and access is lost to the data contained on it. All the data is lost forever unless the drive has been properly maintained. Hard drive maintenance is often neglected. But, it is essential to protecting data and can be performed by a few simple tasks.

The fundamental key to protecting the data contained on a hard disk is to keep a backup at all times. A backup of a hard disk is like a backup of a floppy. It is simply a copy of the data contained on the hard drive. So, if the hard disk crashes you have a copy of the data. A backup is usually kept on floppies, while some systems use a tape. Several floppies are required because of their limited storage capacities.

The process of making a backup can be pure drudgery or quite simple depending on the method used. Using simple copy commands from DOS or the ST GEM Desktop require a lot of careful planning and floppy disk management. It must be decided to which backup floppy each file on the hard disk will be copied. This method is very time consuming and tedious. And as soon as a new file is written to the hard disk or an old one is updated the backup is no longer valid. Further, it must be decided to what floppy these files will be written without starting the entire backup procedure over again.

An easier method of creating a hard disk backup is to use a utility program written specifically for the purpose. Such a utility is Michtron's BACKUP! for the Atari ST. With it, a full backup is created first which is a copy of all the data on the hard drive on a set of floppies. Then incremental backups are made on separate floppies which contain a copy of new or updated data. Incremental backups should be made on a daily basis and usually do not take very much time to perform. However, BACKUP! seems to only work reliably on Atari made hard drives.

An alternative is to use I.B. Computer's Hard Drive Back Up utility. It performs the functions of BACKUP!, works with all hard drive systems, and supports a 5 1/4" floppy drive, such as the I.B. Drive. There are other commercial and public domain backup programs which may work with different hard drive systems, but should be avoided unless it has been tested on a particular system.

Now that the importance of backups and how to make them is known, it is helpful to understand what leads to hard disk failure. Incidentally, there is a difference between a disk failure and a disk crash. A failure is when only some data is lost or garbled. A crash is the worst case failure where all the data is lost.

One way a hard drive can lose data is by software. Software failures are caused by programs or DOSS which contain coding bugs. These can be found on any machine and usually will cause a failure intermittently. Beware of programs which allow a user to

examine and edit disk sectors. They can wipe out a disk in a split second if they are not used correctly. Anyone not familiar with disk structures should never use any such utility. Finally, the most frustrating software problems are caused by programs which intentionally wipe out data on a hard disk. These are known as viruses, worms, you name it. They are written by mentally ill people and usually infest the public domain. However, it has been reported that some commercial software contained data destructive schemes without the software vendor even being aware of the problem. Therefore, there are no 100% sure ways to avoid this burden. A measure of caution is to run any new piece of software from a floppy disk with the hard drive turned off. Take note of any unusual drive activity and check the floppy for errors after running the software.

Another way a hard drive can lose data is by physical misuse and can occur by several means. The surest way to cause a disk crash is to give the hard drive a sudden jolt (dropping something on the desk, moving the drive, etc.). This applies to whether or not the drive is powered up and being used. The reason for this is because the heads in a hard drive do not make contact with the disks (platters). So, when it experiences a jolt the heads make a disastrous collision with the surface of the platters. This problem much worst when the drive is powered because the platters are spinning at a speed of 3600 RPM while the heads remain stationary. One collision under this circumstance and the hard drive usually becomes useless. Before any hard drive is moved, the heads must be parked to avoid damage. This is usually accomplished by running a program from a floppy disk. Parking the heads is the physical moving of the heads in a hard drive to a area over the platters where no data lies below. So, if the heads come in contact with the surface of the platters it won't destroy any data.

The heads of every hard drive should parked at the end of each computing session to avoid yet another potential problem. This problem is known as a power up crash and can occur anytime a hard drive is powered up. What happens is that when the heads are left unparked, they are left above data sectors. The next time the drive is powered up a surge of electrical power reaches the heads, creates a magnetic field, and zaps the data below. This burst of energy occurs in the same way a thump sound is heard from the speakers of a stereo system when it is powered up. Except that it is the precious data on the hard drive that is at risk. By parking the heads, however, the burst occurs over a unused portion of the platters.

There are programs available for the ST which can find and correct problems on a hard drive. One of them is the excellent Hard Disk Sentry by Beckemeyer. This program will test the disk for problems and report them on screen or printer. Then it will attempt to correct the problems. It can even recover files that otherwise would be considered lost. Finally, it will optimize the disk for maximum speed. Optimizing is the process of reorganizing the data on disk so that each file is made continuous instead of fragmented. Fragmenting occurs naturally and cannot be avoided. But, it can lead to disk failures and will slow down the performance of a hard drive.

Hard disk maintenance is essential for the life of the data contained on it. While hard drive utilities are meant to avoid errors, the data should be backed up before using any one of them. Be especially cautious of all public domain utilities. There price is nice, but the results can be far from what is expected. Keep a hard drive routinely maintained and it will keep the data routinely safe.

J A C G

JERSEY ATARI COMPUTER GROUP

8 CRESCENT ROAD

PINE BROOK, NEW JERSEY 07058

Ed Mathews
31 Mill Creek Rd
Bayville NJ
08721
269-3733

BULK RATE
U.S. POSTAGE
PAID
PINE BROOK, NJ
PERMIT # 56

JACG NEWSLETTER VOL. 8, NUM. 10

DECEMBER 1988

EXECUTIVE COMMITTEE

ADVERTISING RATES

Full Page (7.5 x 9) $48.00
Half page $25.00
Quarter Page $18.00

Discount Rates available on request.

J.A.C.G. MEMBERSHIP APPLICATION

DUES: U.S. 3rd Class Mailing, Canada, Mexico $25.00
U.S. 1st Class Mailing, Foreign subscriptions 31.00

___ Renew ___ New ___ Former
___ 8-Bit ___ 16-Bit

NAME _____
Address _____
City _____
State/ Country/ Zipcode _____
Home Phone Number _____
Date _____ 1st Class _____ 3rd Class _____

MAIL TO: Robert Mulhearn
8 Crescent Rd, Pinebrook, NJ 07058